Semestrální práce z předmětu KIV/DS

ZeroMQ služba s integrovaným Chandy-Lamportovým algoritmem

Martin Matas
A18N0095P
martinm@students.zcu.cz

22. ledna 2020

Obsah

1 Zadání

Implementace ZeroMQ služby s integrovaným Chandy-Lamportovým algoritmem pro získání konzistentního snímku globálního stavu.

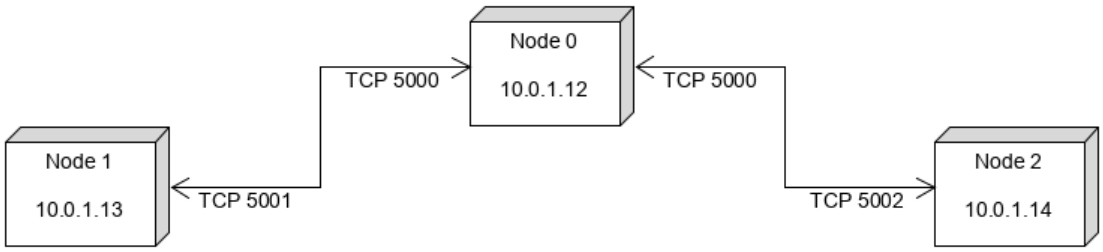
- Implementujte bankovní službu s operacemi (lze využít implementaci z předchozí úlohy)
  - credit – přičtení částky k zůstatku na účtu
  - debit – odečtení částky od zůstatku na účtu

  tak, že obě operace lze provést na základě zprávy doručené do vstupní ZeroMQ fronty následovně: pokud klient chce poslat částku do jiné banky, odečte částku a pošle zprávu CREDIT s částkou do jiné banky (fronty). Pokud chce klient inkasovat částku z jiné banky, pošle pouze zprávu DEBIT s částkou, přičemž druhá strana částku odečte a pošle ji ve zprávě CREDIT.
- Všechny uzly jsou identické, kromě IP adresy a jména uzlu, jejich počet je v rozmezí $2 - 5$
- Každý uzel po startu začíná se zůstatkem na účtu 5.000.000.
- Uzly náhodně vyberou jednu z operací CREDIT či DEBIT s náhodnou částkou v intervalu $< 10.000, 50.000 >$ a odešlou zprávu s operací na náhodně vybraný sousední uzel. Pokud uzel na svém účtu nemá dostatečnou částku, operaci odmítne.
- Uzly spolu komunikují pouze přes vstupní ZeroMQ fronty (podle vzoru PAIR), které představují komunikační kanály. Zprávy přenášejte v JSON formátu.
- Topologii si zvolte sami, ale je nutné ji uvést v průvodní dokumentaci.
- Jako servisní službu implementujte Chandy-Lamportův distribuovaný algoritmus pro získání konzistentního snímku globálního stavu podle stejného principu jako bankovní API (zpráva "marker" ve vstupní frontě – viz popis algoritmu).
- Spuštění Chandy-Lamportova algoritmu zahajte posláním zprávy MARKER do libovolné fronty (z příkazové řádky na libovolném uzlu).
- Ošetřete i případ, že může být v jednom okamžiku spuštěno více snapshotů.
- Stavem uzlu se rozumí zůstatek na účtu v okamžiku přijetí prvního markeru, stavem kanálu seznam zpráv s bankovními operacemi přijatými po prvním markeru (viz popis algoritmu).
- Jakmile detekujete ukončení algoritmu, všechny uzly pošlou svoje uložené stavy do vyhrazené fronty, ze které je vyhrazený proces vybírá a zobrazuje (stačí na konzoli).

- K vytvoření infrastruktury použijte nástroje Docker a Kubernetes (Kubelet), ale je možné použít i Vagrant.
- Aplikaci můžete implementovat v Jazyce Java nebo Python s využitím již existujícího software, který vám usnadní implementaci jednotlivých modulů a jejich vzájemnou komunikaci.
- Zdrojové kódy udržujte a publikujte v repozitáři https://gitlab.kiv.zcu.cz

2 Analýza

Dle zadání bude pro komunikaci mezi jednotlivými uzly sítě použita služba `ZeroMQ` a pro získání globálního stavu bude implementován Chandy-Lamportův algoritmus. Pro implementaci serveru se nabízí využít programovací jazyk Java pro jeho snadnou práci s vlákny a kritickými sekcemi. Pro simulaci bankovní sítě byla navržena následující topologie uvedená na obrázku 1, kde popisy na hranách představují port, na kterém daný uzel naslouchá.



Obrázek 1: Topologie sítě.

3 Implementace

Server byl implementován v jazyce `Java 8` s využitím knihovny `Jeromq` pro implementaci služby ZeroMQ. Implementace serveru se dále pro přehlednost dělí do 4 následujících balíků a spustitelné třídy pro spuštění serveru:

config - načtení konfiguračního souboru s topologií sítě

domain - třídy zastupující potřebné objekty např. bankovní účet, zprávy, atd.

util - nástroje pro konverzi dat, úložiště globálního stavu uzlu

zmq - implementace pro zasílání a příjímání zpráv, obsluha požadavků

Server.java - spustitelná třída, která vytváří pool vláken pro workery

Při spuštění aplikace nejprve dojde k načtení konfigurace topologie a následně k inicializaci dynamického poolu vláken, tzn. aplikace má tolik vláken, kolik je dohromady instancí třídy `Sender` a `Listener` (chcete-li celkový počet odchozích a příchozích kanálů). Je tomu tak především kvůli častým problémům se zpracováním příchozích zpráv z více než jednoho klienta, kdy zprávy nebyly ani přijaty.

Výše zmíněné třídy `Sender` a `Listener` jsou `Runnable` implementace pro odesílání a příjem zpráv ve formátu JSON. Třída `Sender` poskytuje jak statické metody pro odpovědi na příchozí zprávy, tak nekonečnou smyčku s příkazy pro generování bankovních požadavků. Třída `Listener` pak slouží pro naslouchání na definovaných portech a pro zpracování a obsluhu příchozích požadavků.

Pro přehlednost byla zavedena pravidla definování portů, kdy pro běžnou komunikaci (`CREDIT`, `DEBIT`, `MARKER` zprávy) slouží porty ve tvaru **500x**, pro zahájení snímku globálního stavu pak ve tvaru **5555** a nakonec ještě pro příjem a výpis globálního stavu (zpráva typu `GLOBAL_STATE`) jsou porty definovány ve tvaru **555x**, kde **x** je číslo uzlu sítě. Opět je důvodem zmíněný problém, kdy jeden kanál nebyl schopen přijímat zprávy z více klientů.

3.1 Globální stav

Pro zachycení globálního stavu byla implementována třída `LocalStateLogger`, která umožňuje zachycení aktuálního stavu bankovního účtu, záznam příchozích zpráv, výpis zaznamenaného stavu a kontrolu nad procesem logování změn.

Pro možnost provádět vícero snímků, byl proces sledování globálního stavu ošetřen unikátním identifikátorem, který zajišťuje jednoznačnou identifikaci konkrétního globálního stavu.

Proces zachycení globálního stavu se spouští odesláním zprávy z klienta na uzel s označením `Node 0` na adresu `10.0.1.12:5555`.

4 Uživatelská příručka

V kořenovém adresáři se nachází soubor `Vagrantfile`, který popisuje infrastrukturu a dostupné nástroje jednotlivých uzlů. Pro spuštění serverů je potřeba mít nainstalovaný nástroj `Vagrant`, poté stačí použít příkaz `vagrant up` v kořenovém adresáři repozitáře. Uzly si sami stáhnou vše potřebné pro běh a následně přeloží, sestaví a spustí serverovou aplikaci.

Zachycení snímku globálního stavu se provádí skriptem, který je umístě v kořenovém adresáři pod názvem `take_snapshot.sh`. Pro jeho spuštění bude potřeba mít nainstalovaný `Python 2` a správce balíků `pip`. Skript při spuštění nejprve doinstaluje balíček **pyzmq** a poté spustí program, který odešle zprávu na server, čímž zahájí

proces snímání globálního stavu. Po skončení bude výsledek globálního stavu vypsán v serverovém logu `server.log` v kořenovém adresáři.

V souboru `bankserver/src/main/resources/nodes.yaml` jsou definovány konfigurace jednotlivý uzlů sítě a jejich komunikační porty společně se sousedními uzly.

5 Závěr

Implementoval jsem bankovní službu, která pro komunikaci využívá ZeroMQ s integrovaným Chandy-Lamportovým algoritmem pro získání konzistentního snímku globálního stavu. Při implementaci jsem se až na komplikace s příjmem zpráv z více klientů nepotýkal s žádnými problémy.